SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2021, Vol. 64 122101:1–122101:15
https://doi.org/10.1007/s11432-020-3024-5

Neural compositing for real-time augmented reality rendering in low-frequency lighting environments

Shengjie MA[1†], Qian SHEN[1†], Qiming HOU[1,2], Zhong REN[1,2] & Kun ZHOU[1,2*]

[1] *State Key Lab of CAD&CG, Zhejiang University, Hangzhou* 310058*, China;*
[2] *ZJU-FaceUnity Joint Lab of Intelligent Graphics, Hangzhou* 310015*, China*

Received 13 June 2020/Revised 13 July 2020/Accepted 29 July 2020/Published online 5 January 2021

Abstract We present neural compositing, a deep-learning based method for augmented reality rendering, which uses convolutional neural networks to composite rendered layers of a virtual object with a real photograph to emulate shadow and reflection effects. The method starts from estimating the lighting and roughness information from the photograph using neural networks, renders the virtual object with a virtual floor into color, shadow and reflection layers by applying the estimated lighting, and finally refines the reflection and shadow layers using neural networks and blends them with the color layer and input image to yield the output image. We assume low-frequency lighting environments and adopt PRT (precomputed radiance transfer) for layer rendering, which makes the whole pipeline differentiable and enables fast end-to-end network training with synthetic scenes. Working on a single photograph, our method can produce realistic reflections in a real scene with spatially-varying material and cast shadows on background objects with unknown geometry and material at real-time frame rates.

Keywords augmented reality, neural networks, differentiable renderer, reflection, shadow

Citation Ma S J, Shen Q, Hou Q M, et al. Neural compositing for real-time augmented reality rendering in low-frequency lighting environments. Sci China Inf Sci, 2021, 64(2): 122101, https://doi.org/10.1007/s11432-020-3024-5

1 Introduction

The availability of commercial AR (augmented reality) platforms like ARKit[1)] and ARCore[2)] enables application developers to render virtual objects into real scenes with just a smart phone. Current tracking technologies estimate camera poses and localize real-world planes with considerable accuracy. However, making the rendered objects appear real is still far from trivial.

A key challenge is the interaction between virtual objects and the real scene. Virtual objects can cast spatially-varying shadow and produce reflection on real ones, like in Figure 1. Such illumination effects are crucial to realism, yet difficult to simulate because it is hard to accurately reconstruct the geometry and material information of the real scene from limited input images.

In this paper, we introduce neural compositing, a real-time method to render a virtual object into a real photograph with reflection and shadow effects. Inspired by the production practice of digital compositors, our method uses CNNs (convolutional neural networks) to composite rendered layers of the virtual object with the photograph to emulate effects expensive to render physically. The method starts from estimating the lighting and roughness information from the photograph using CNNs, renders the virtual object with a virtual floor into color, shadow and reflection layers by applying the estimated lighting, and finally refines the reflection and shadow layers with CNNs and blends them with the color layer and input image to yield the output image. We assume low-frequency lighting environments approximated by SH (spherical harmonics) functions and adopt PRT (precomputed radiance transfer) [1] for layer rendering, which makes the whole pipeline differentiable and enables fast end-to-end network training with synthetic

* Corresponding author (email: kunzhou@zju.edu.cn)
† Ma S J and Shen Q have the same contribution to this work.
1) ARKit. Version 3.5. Apple. 2020.
2) ARCore. Version 1.16.0. Google. 2020.

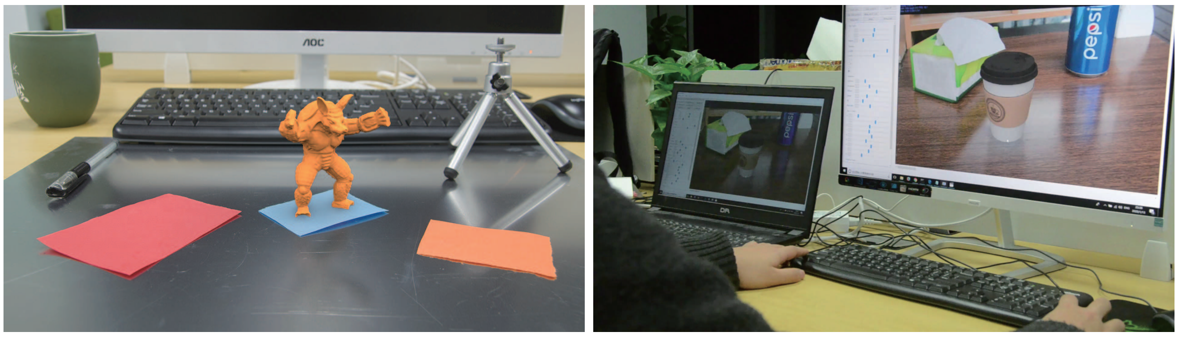
Figure 1 (Color online) Our real-time AR system. Left: a virtual armadillo rendered into a photograph. The system correctly shows reflection on the table while only casting shadow on the blue paper. Right: a user manipulating a virtual coffee cup on a laptop at 25 fps.

images. Working on a single photograph, our method can produce plausible reflections in a real scene with spatially-varying material and cast shadows on background objects with unknown geometry and material at real-time frame rates. We provide the supplementary video[3] to show our results.

In summary, our paper makes the following contributions.

(1) We introduce a neural compositing method for AR rendering that uses CNNs to combine rendered layers of virtual objects with real images to emulate physically sophisticated effects, sidestepping accurate reconstruction of geometry and material required by many previous methods.

(2) We propose to use PRT for layer rendering, yielding a differentiable AR rendering pipeline that can be fast trained end-to-end on a synthetic dataset.

(3) We present an AR system that renders virtual objects into single-view photographs with realistic shadow and reflection effects in low-frequency lighting environments at real-time frame rates.

2 Related work

Compositing a 3D object into a 2D photograph is a well-researched topic. One class of methods is harmonization by adjusting brightness and hue of foreground to match the scene. Existing methods include matching statistics [2–4], gradient-based blending [5, 6], learning-based methods [7–10]. These methods do compositing in an end-to-end manner without explicit reconstruction of lighting, geometry and albedo. However, physical correctness is not warranted and virtual objects' effects on the scene like shadow and reflection are totally ignored which are critical in AR applications. Another class of established methods create realistic results using differential rendering [11], where a reconstruction of the real scene is rendered with and without the virtual objects to generate a differential image additively blended onto the photograph.

The reconstruction can be performed by photographing light probes [11, 12], labeling lights interactively [11], or be automated with an optimization process [13–15]. Deep neural networks can also learn relevant information from photographs, including lighting [16–20], geometry and albedo [21], or even SVBRDF [22–25]. Regardless of how the real scene is reconstructed, though, final compositing still comes to differential rendering [12, 13, 18, 26–28]. As a result, any rendering artifacts directly go into the final result, placing a high quality demand on the reconstructed scene.

Our method approaches the AR rendering problem from a different angle by directly improving the final compositing. Specifically, we use neural networks to post-process challenging shadow and reflection layers to hide artifacts and match the photograph, before blending them onto the final result. This allows us to significantly weaken the quality requirement of the rendered images, sidestepping challenging reconstruction tasks. The inspiration comes from the production practice of digital compositors, who enhance renderer output in image processing software for the same purpose.

A different direction is neural rendering, which replaces most or all of the shading process with neural networks [29, 30]. Such methods can synthesize realistic appearance from latent, semantic or geometric features, an attractive feature when handling real or reconstructed objects. However, this feature does not

3) https://zjumsj.github.io/NCRTAR.github.io/.

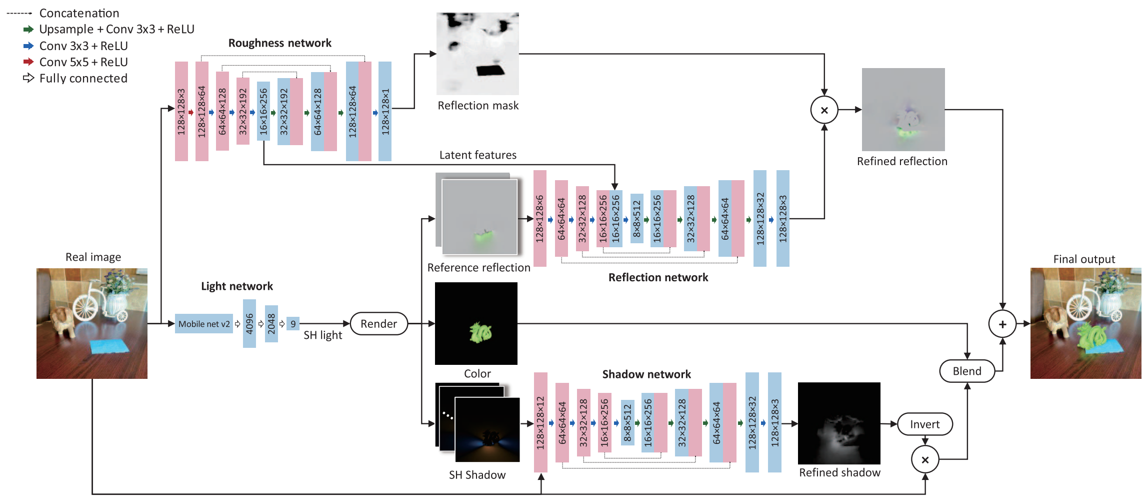

Figure 2 (Color online) The pipeline of our method and network architectures. Some images have been enhanced to improve legibility. The pipeline is composed of three U-Nets [37] like networks (roughness network, reflection network and shadow network), a light network resembling DeepLight [18], a conventional real-time renderer and several fixed-function blending steps.

apply as well to virtual objects, where authored materials make latent models less necessary. Our post-processing networks are analogous to neural renderers. However, the input to our networks is reference shadow and reflection layers produced by a conventional real-time renderer instead of raw features.

Differentiable rendering is an essential utility when training neural networks for graphics tasks, and our system requires one such renderer. There are many fully generic frameworks offering differentiability for all scene parameters [31–33]. However, their per-iteration Monte Carlo sampling is too expensive for our application which does not require such generality. While non-physical light transport approximations can be made for significant acceleration [34, 35], they also diverge from our training target where physical correctness is critical. Instead, we focus on light differentiability and use PRT [1] to reduce light transport to a differentiable linear process while maintaining physical correctness, analogous to the inverse cloud rendering framework of Dobashi et al. [36].

3 The method

3.1 Problem setting

Our method works on a single image of the real world. We assume the AR platform has provided a corresponding camera pose and located an infinitely large plane which we call the floor. Virtual objects are positioned on the floor. To disambiguate object albedo from light intensity, we assume the floor has a fixed albedo of 0.5 and the lighting is monochrome, which has a side benefit of providing a reference for virtual materials. We further assume the virtual objects, or foreground objects, are positioned in front of all real objects, also referred as background objects in the image.

3.2 Overview

As illustrated in Figure 2 [18, 37], our algorithm consists of four neural networks, one rendering step, and a few fixed-function blending steps. Given the real image, we first estimate the lighting and roughness information with the light network and roughness network respectively. The reconstructed low-frequency SH lighting is then used to render the virtual object and a virtual floor into a color layer, shadow layers, and reflection layers for three predefined reference floor materials, one diffuse, one glossy and one specular.

The reflection network is used to combine the rendered reflection layers on predefined materials with latent features extracted by the roughness network to synthesize a refined reflection layer where the sharpness is consistent with that in the real image. The shadow network is used to combine the rendered shadow layers with the input image to correct for non-floor objects, removing shadows occluded by them while adding shadows for objects that appear close to the virtual one. Finally the refined layers and the color layer are blended with the input image to generate the final output. The intermediate and final

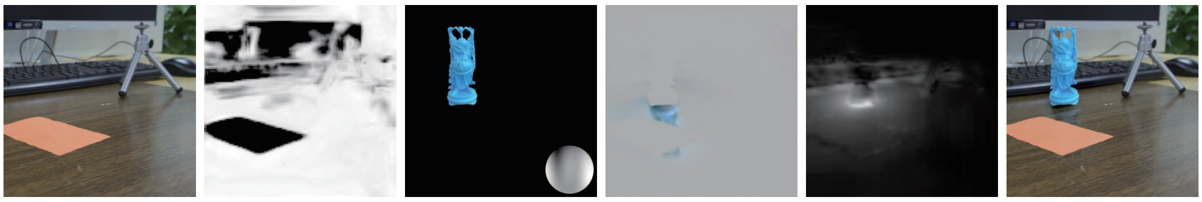

Figure 3 (Color online) Intermediate results in a compositing process. From left to right: the input image, the reflection mask, the color layer (with the estimated SH lighting shown as inset), the refined reflection layer, the refined shadow layer, and the final output image.

results of one complete compositing are visualized in Figure 3 to give our readers an intuition on how the pipeline works.

The neural networks are trained end-to-end on a synthetic dataset. We use PRT to render the color, shadow and reflection layers that are taken by the networks as input. The rendering step is reduced to simple linear combinations, and thus it is differentiable.

3.3 Blending and rendering

The final blending steps in Figure 2 are the key inspiration we take from production practice. Mathematically, given the real image as a background B, a shadow layer S, a reflection layer R, a color layer C with RGB components and an alpha component A, the final output I is computed as

$$I = B \circ (1 - S) \circ (1 - A) + C \circ A + R, \tag{1}$$

where $\circ$ indicates element-wise multiplication. This process matches the compositing network that artists often employ in off-the-shelf software. The shadow term masks out the background through a multiplication of $1 - S$. C and A are alpha-blended onto the shadowed background and the reflection term R is simply added at the end.

Strictly speaking, Eq. (1) is not physically correct. The optical effects casted by virtual objects in the real image are far more complicated than just subtracting a shadow and adding a reflection. The equation is more of a best-effort approximation that turns out to work well in practice, especially when the S and R layers were manually tweaked. For example, when a virtual shadow needs to be casted onto a real object, an artist could rotoscope a new patch of shadow and add it to S. We seek to replicate such a compositing process with our neural networks.

The rendering step and the two following networks have to produce layers that eventually become meaningful when substituted into (1). The color and alpha layers are straightforward to produce and we will focus the discussion on the shadow and reflection layers. Consider a point x on the floor, we denote its radiance in the real image as $B(x)$ and its radiance after rendering the virtual object as $I(x)$. Since x lies on the floor, we ignore foreground color C and alpha A to simplify (1) into

$$I(x) = B(x)\,(1 - S(x)) + R(x), \tag{2}$$

where S and R correspond to the shadow and reflection layers respectively.

The above terms can be written as hemispherical integrations:

$$B(x) = \int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega) \mathrm{d}\omega, \tag{3}$$

$$I(x) = \int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega)\,(1 - S_{\mathrm{F}}(x,\omega))\,\mathrm{d}\omega + R(x), \tag{4}$$

$$R(x) = \int L_{\mathrm{F}}(x,\omega) H_{\mathrm{B}}(x,\omega) S_{\mathrm{F}}(x,\omega) \mathrm{d}\omega, \tag{5}$$

where subscripts B and F indicate background and foreground terms respectively. $L_{\mathrm{B}}(x,\omega)$ is the real-world radiance at point x from direction ω, H_{B} is the floor BRDF term, and V_{B} refers to real-world visibility. $S_{\mathrm{F}}(x,\omega)$ is a foreground shadow term that is 1 if an ray from point x along direction ω would intersect a foreground object and 0 otherwise. $L_{\mathrm{F}}(x,\omega)$ is the radiance from foreground objects when seen from point x along direction ω, i.e., the inter-reflection. At a high level, Eq. (4) simply decomposes the post-AR radiance into a shadowed background integration term and a reflected foreground term $R(x)$.

To derive $S(x)$, we substitute (4) into (2):

$$\begin{aligned} S(x) &= 1 - \frac{I(x) - R(x)}{B(x)} \\ &= 1 - \frac{\int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega)(1 - S_{\mathrm{F}}(x,\omega))\mathrm{d}\omega}{\int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega)\mathrm{d}\omega} \\ &= \frac{\int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega}{\int L_{\mathrm{B}}(x,\omega) H_{\mathrm{B}}(x,\omega) V_{\mathrm{B}}(x,\omega)\mathrm{d}\omega}. \end{aligned} \tag{6}$$

Note that L_{B}, H_{B} and V_{B} are real-world values thus unavailable to the renderer, and we have to provide substitutions. Utilizing the PRT [1] formulation, we approximate $L_{\mathrm{B}}(x,\omega)$ with a global lighting represented as order 3 SH $\tilde{L}_{\mathrm{B}}(\omega) = \sum_{l<3} \tilde{c}_{l,m} Y_{l,m}(\omega)$ where $Y_{l,m}(\omega)$ is the SH basis functions, and $\tilde{c}_{l,m}$ is the SH coefficients reconstructed by the light network. The foreground radiance $L_{\mathrm{F}}(x,\omega)$ can be itself written as the dot product of incident lighting $\tilde{L}_{\mathrm{B}}$ and a transfer function $T_{\mathrm{F}}(x,\omega',\omega)$. For H_{B}, we render three reference materials $\tilde{H}_{\mathrm{diff}}$, $\tilde{H}_{\mathrm{glossy}}$, $\tilde{H}_{\mathrm{spec}}$, which are respectively a Lambertian BRDF, a glossy GGX BRDF [38] and a specular BRDF. All reference materials use the assumed floor albedo of 0.5. And we simply ignore V_{B}. Substituting the approximations into (5) and (6) yields

$$\begin{aligned} \tilde{R}_i(x) &= \int L_{\mathrm{F}}(x,\omega) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega \\ &= \int \left(\int \tilde{L}_{\mathrm{B}}(\omega_i) T_{\mathrm{F}}(x,\omega_i,\omega)\mathrm{d}\omega_i \right) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega \\ &= \iint \tilde{L}_{\mathrm{B}}(\omega_i) T_{\mathrm{F}}(x,\omega_i,\omega) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega_i\mathrm{d}\omega \\ &= \sum \tilde{c}_{l,m} \iint Y_{l,m}(\omega_i) T_{\mathrm{F}}(x,\omega_i,\omega) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega_i\mathrm{d}\omega, \end{aligned} \tag{7}$$

$$\begin{aligned} \tilde{S}_{\mathrm{diff}}(x) &= \frac{\int \tilde{L}_{\mathrm{B}}(\omega) \tilde{H}_{\mathrm{diff}}(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega}{\int \tilde{L}_{\mathrm{B}}(\omega) \tilde{H}_{\mathrm{diff}}(x,\omega)\mathrm{d}\omega} \\ &= \frac{\sum \tilde{c}_{l,m} \int Y_{l,m}(\omega) \tilde{H}_{\mathrm{diff}}(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega}{\sum \tilde{c}_{l,m} \int Y_{l,m}(\omega) \tilde{H}_{\mathrm{diff}}(x,\omega)\mathrm{d}\omega}, \end{aligned} \tag{8}$$

$$\begin{aligned} \tilde{S}_i(x)B(x) &= \int \tilde{L}_{\mathrm{B}}(\omega) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega \\ &= \sum \tilde{c}_{l,m} \int Y_{l,m}(\omega) \tilde{H}_i(x,\omega) S_{\mathrm{F}}(x,\omega)\mathrm{d}\omega, \end{aligned} \tag{9}$$

where $i \in \{\mathrm{glossy}, \mathrm{spec}\}$.

The $\tilde{R}(x)$ and $\tilde{S}(x)$ terms are what the renderer produces and what the neural networks take as input. $\tilde{S}_i(x)B(x)$ signifies the absolute radiance shadowed by foreground objects and is also taken as the networks' input (detailed in Subsection 3.4). Since $\tilde{c}_{l,m}$ are the only terms dependent on background, we can pre-compute the $Y_{l,m}$-related terms, which correspond to rendering the layers under SH basis lighting:

$$\tilde{R}_i(x) = \sum \tilde{c}_{l,m} \tilde{R}_i^{l,m}(x), \tag{10}$$

$$\tilde{S}_{\mathrm{diff}}(x) = \frac{\sum \tilde{c}_{l,m} \tilde{S}_{\mathrm{diff,n}}^{l,m}(x)}{\sum \tilde{c}_{l,m} \tilde{S}_{\mathrm{diff,d}}^{l,m}(x)}, \tag{11}$$

$$\tilde{S}_i(x)B(x) = \sum \tilde{c}_{l,m} \tilde{S}_i^{l,m}(x). \tag{12}$$

Following the PRT framework, our renderer pre-computes the SH layers on the right side of (10)–(12) and only needs to perform linear combination during interactive rendering and network training. For reflection, the interactive renderer simply renders a mirror image of the virtual objects as $\tilde{R}_{\mathrm{spec}}$ and $\tilde{S}_{\mathrm{spec}}B(x)$. The mirror image rendering is then projected onto a camera-facing billboard perpendicular to the floor. The GGX reflection of this billboard is sampled in a shader to approximate the glossy layers

$\tilde{R}_{\text{glossy}}$, $\tilde{S}_{\text{glossy}}B(x)$. For diffuse shadow, the layer is split into two pre-computed terms, $\tilde{S}_{\text{diff,n}}$ and $\tilde{S}_{\text{diff,d}}$, which are referred as numerator and denominator from now on.

Discussion. Note that while more accurate approximations are possible, the current over-simplified method is intentionally chosen. We assume that the neural networks can handle the more advanced effects we approximate away better. Also, while the actual denominator of (8) is simply $B(x)$ and can be fetched precisely from the background image, we choose to use an approximate version for better consistency with the numerator which leads to improved generalization.

3.4 Network architectures

To refine the approximate renderings $\tilde{R}_i$ and $\tilde{S}_i$, we define a shadow network σ_s, a reflection network σ_r and a roughness network σ_m that computes reflection mask M and latent features F. The SH lighting coefficients $\tilde{c}_{l,m}$ in Subsection 3.3 also need to be reconstructed using a light network σ_l. The role of our neural networks are summarized as

$$c_{l,m} = \sigma_l(w_l, B), \tag{13}$$

$$S = \sigma_s\left(w_s, B, \frac{\tilde{c}_{l,m}\tilde{S}^{l,m}_{\text{diff,n}}}{\sum \tilde{c}_{l,m}\tilde{S}^{l,m}_{\text{diff,d}}}\right), \tag{14}$$

$$\begin{pmatrix} M \\ F \end{pmatrix} = \sigma_m(w_m, B) \tag{15}$$

$$R = M \circ \sigma_r(w_r, F, \tilde{R}_{\text{glossy}} - \tilde{S}_{\text{glossy}} \circ B, \tilde{R}_{\text{spec}} - \tilde{S}_{\text{spec}} \circ B), \tag{16}$$

where w_l, w_s, w_f and w_r are network weights. The $\circ$ operator in the specular and glossy shadow terms $\tilde{S}_i \circ B$ is not actually evaluated in our implementation as $\tilde{S}_i \circ B$ can be directly calculated by (12).

We combine shadow layers $\tilde{S}_{\text{glossy}} \circ B$ and $\tilde{S}_{\text{spec}} \circ B$ with respective reflection layers and send them to the reflection network σ_r because the appearance of said layers resembles specular reflection $\tilde{R}_{\text{spec}}$ more than diffuse shadow $\tilde{S}_{\text{diff}}$. The roughness network σ_m is factored out of σ_r as an optimization since it only involves the background image B and is relatively expensive to evaluate. The SH layers $\tilde{S}^{l,m}_{\text{diff,n}}$ are sent to the shadow network σ_s individually after multiplying lighting coefficients and dividing the combined denominator, and their combination is performed by the network. We found this input layout generalizes better than just passing one combined shadow layer.

Figure 2 illustrates our network architectures. We use the encoder part of DeepLight [18] for light estimation and replace their decoder with a few fully-connected layers to generate SH coefficients instead of a full environment map. The roughness, shadow and reflection networks resemble U-Nets [37]. All color images involved using the linear color space and appropriate conversion is done during acquisition of B and display of I. To handle input with higher resolution than the input size of 128×128, we downsample the network input layers and upsample their output layers to the original resolution with bilinear interpolation.

3.5 Training dataset

We train our networks on a set of $\sim$80000 synthetic images. We generate training scenes with randomized geometry, material, lighting and camera poses, and then render them to yield training images. Figure 4 illustrates some example scenes used in training.

We choose to focus on diversity and physical correctness instead of visual realism when designing our training data. The motivation is two-fold. First, synthesizing a realistic scene is an open problem with its own challenges which we wish to avoid. Instead, we seek to randomize in a “superset of reality”, i.e., an exaggerated setting more diversified than the real world, in the hope that the trained network would generalize to its realistic subset. Second, our training process takes advantage of additional output layers like isolated reflection. Such layers are difficult to obtain from a real photograph while trivial to synthesize in a renderer. Under this design philosophy, we created a simple set of randomization rules which we will detail in the following.

Geometry. Our random scene consists of an infinite plane for the floor and 2 to 4 randomly generated objects placed on top of it. Each object is represented by a zero-clamped combination of SH order 5

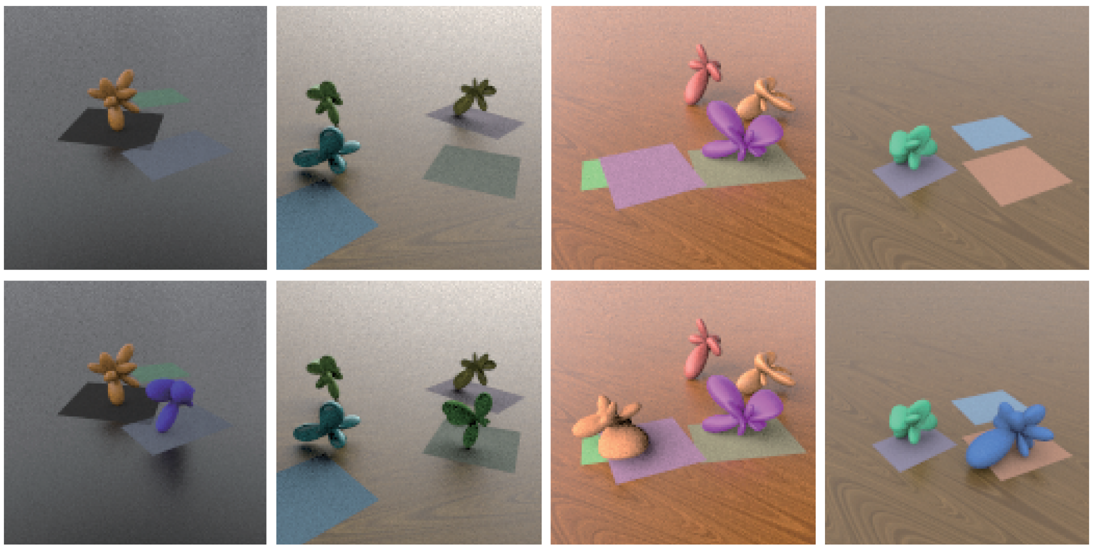

Figure 4 (Color online) Example training scenes. We use image resolution of 128 × 128. Background images are shown in the upper row and final images are shown in the lower one.

basis with uniformly distributed random coefficients, used as a spherical height map. More precisely, each object is a point set in the following form:

$$\left\{ \max\left(0, \sum_{l<5} x_{l,m} Y_{l,m}\left(\omega\right)\right) \omega \,\middle|\, x_{l,m} \sim U(-1,1), \omega \in \Omega \right\}, \tag{17}$$

where U refers to the uniform distribution. All length units are millimeters.

Scenes with overlapped object bounding boxes are rejected. After all objects in a scene are generated, the one closest to the camera is selected as the foreground object, which corresponds to the virtual object in AR rendering. All other objects and the floor form the background, which corresponds to the real photograph.

Material. For all objects, we use a GGX BRDF combined with a Lambertian BRDF using a Fresnel term and only randomize the roughness. All surfaces use a procedural wood texture4), which we modify to exaggerate distortion and replace the original wood-like colors with random, saturated ones. Empirically, we find that networks trained with more saturated colors and more distorted patterns generalize better.

To emulate spatial variance, we place 0–3 random rectangular patches on the floor. Each patch has a random material different from the floor itself. Also, while we assume the floor has a fixed albedo when rendering, the training scenes have floor albedo randomized in a small range 0.3–0.5 to improve generalization.

Lighting. We use the HDR environment map dataset of Gardner et al. [39] and project the images to SH order 3, matching the output of our light network. Each training scene is illuminated by combining 1–3 such SH environment maps, each with independently generated random rotation and intensity. Note that linear combination of rotated and scaled SH lobes does not change the order, and the final lighting can still be represented in SH order 3.

Camera pose. We use a camera that looks at the origin with zero roll angle from a random point in space. Assuming the floor plane is XY and the $Z+$ axis points up, the camera point is uniformly sampled within the cube $(-50, 50) \times (-1300, -1100) \times (600, 700)$. The camera FOV is uniformly sampled within the range $(46°, 48°)$, which resembles typical square pictures taken by smart phones.

Rendering. We built a custom physically-based path tracer to render the generated scenes. Each scene is rendered in three passes, one with background objects only (noted as "BG"), one with the foreground object and a placeholder floor (noted as "FG"), and the last one with all objects combined (noted as "ALL"). To facilitate pre-training steps that will be explained in Subsection 3.6, the render produces several layers in addition to the final image, as listed in Table 1.

Most layers direct correspond to equation terms in Subsection 3.3. In the FG pass, the floor is a placeholder object that intersects rays but does not have a fixed material. All three reference materials

4) Procedural solid wood. Antovsky. Retrieved from Shadertoy BETA.

Table 1 Rendered layers[a)]

Pass	Term	Description
BG	B	Background layer
FG	A	Foreground alpha
FG	$\tilde{C}^{l,m}$	Foreground color under SH basis lighting $Y_{l,m}$
FG	$\tilde{R}^{l,m}_{\text{spec}}$	Specular reflection under $Y_{l,m}$
FG	$\tilde{R}^{l,m}_{\text{gloss}}$	Glossy reflection under $Y_{l,m}$
FG	$\tilde{S}^{l,m}_{\text{spec}} \circ B$	Specular shadow under $Y_{l,m}$
FG	$\tilde{S}^{l,m}_{\text{gloss}} \circ B$	Glossy shadow under $Y_{l,m}$
FG	$\tilde{S}^{l,m}_{\text{diff,n}}$	Diffuse shadow numerator under $Y_{l,m}$
FG	$\tilde{S}^{l,m}_{\text{diff,d}}$	Diffuse shadow denominator under $Y_{l,m}$
ALL	I	Ground truth final image
ALL	R	Ground truth floor reflection
ALL	S	Ground truth shadow
ALL	M	Pre-training reflection mask

a) BG refers to the background pass, FG refers to the foreground pass, ALL refers to the combined pass.

(diffuse, glossy and specular) are sampled when it is hit by a camera ray and non-camera-ray hits are dropped. Reflection layers are rendered by filtering camera-floor-foreground paths with a path expression. The shadow layers are computed by sampling the light probe directly on floor-reflected rays that intersect the foreground object. The pre-training reflection mask M is approximated as one minus the GGX roughness.

In each render pass, all layers are computed from the same set of path samples to ensure consistency. These include layers that could have been computed analytically like $\tilde{S}^{l,m}_{\text{diff,d}}$.

For dataset generation, we develop an offline renderer based on the Intel Embree framework [40]. We use resolution of 128×128 and sample 512 paths per pixel per pass. The layers are saved as multi-channel OpenEXR[5)] files using its default compression setting.

3.6 Training

The use of PRT makes our entire pipeline differentiable with respect to network weights, since the rendering step only involves simple linear combination as in (10)–(12). This facilitates end-to-end training. To accelerate convergence, though, we also pre-train individual networks before putting them together. Using synthetic data allows us to generate ground truth values of intermediate output, which significantly boosts the pre-training performance.

We start by pre-training the light network. For each scene, we save the ground truth light coefficients $c_{l,m}$ and use them as the target label. Then we pre-train the shadow network by feeding the ground truth light coefficients and reflection into the blending pipeline to make the network concentrate on shadows. The loss is still computed on the final image I.

The training of roughness and reflection networks is more sophisticated. First, we train the roughness network up to the latent feature layer and the reflection network using image loss, with the shadow layer replaced with ground truth and the reflection mask fixed to all-ones. Then we train the mask part of the roughness network in isolation using the latent features generated in the previous step as input, and compute the loss between the mask output and the pre-training mask M. Discrepancy between the inverted roughness and the actual reflection mask is corrected during the final end-to-end training.

We use TensorFlow to train our models using Adam [41] optimizer with a learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a batch size of 16 in both pre-training stage and final end-to-end training stage. We use L1 loss for pre-training light network with light coefficients and Laplacian pyramid L1 [42] loss for image space loss. We find Laplacian pyramid loss works better than naive pixel-wise loss. We train our models on a single NVIDIA GTX 1080Ti GPU. It takes about 20 h for pre-training the light network, 10 h for shadow network, 14 h for the front part of roughness network as well as reflection network and 6 h for the mask part of the roughness network. The final end-to-end training takes about 10 h.

5) OpenEXR. Version 2.4.0. Academy Software Foundation. 2019.

Table 2 Quantitative comparisons of L1 and L2 loss on synthetic dataset[a)]

Loss	All effects	Raw shadow	Raw refl.
L1	**0.0088**	0.0301	0.0135
L2	**0.0247**	0.0840	0.0384

a) "All effects" runs the full pipeline. "Raw shadow" replaces refined shadow with the raw shadow layer. "Raw refl." simply uses a specular reflection instead of refined reflection. The best scores are highlighted.

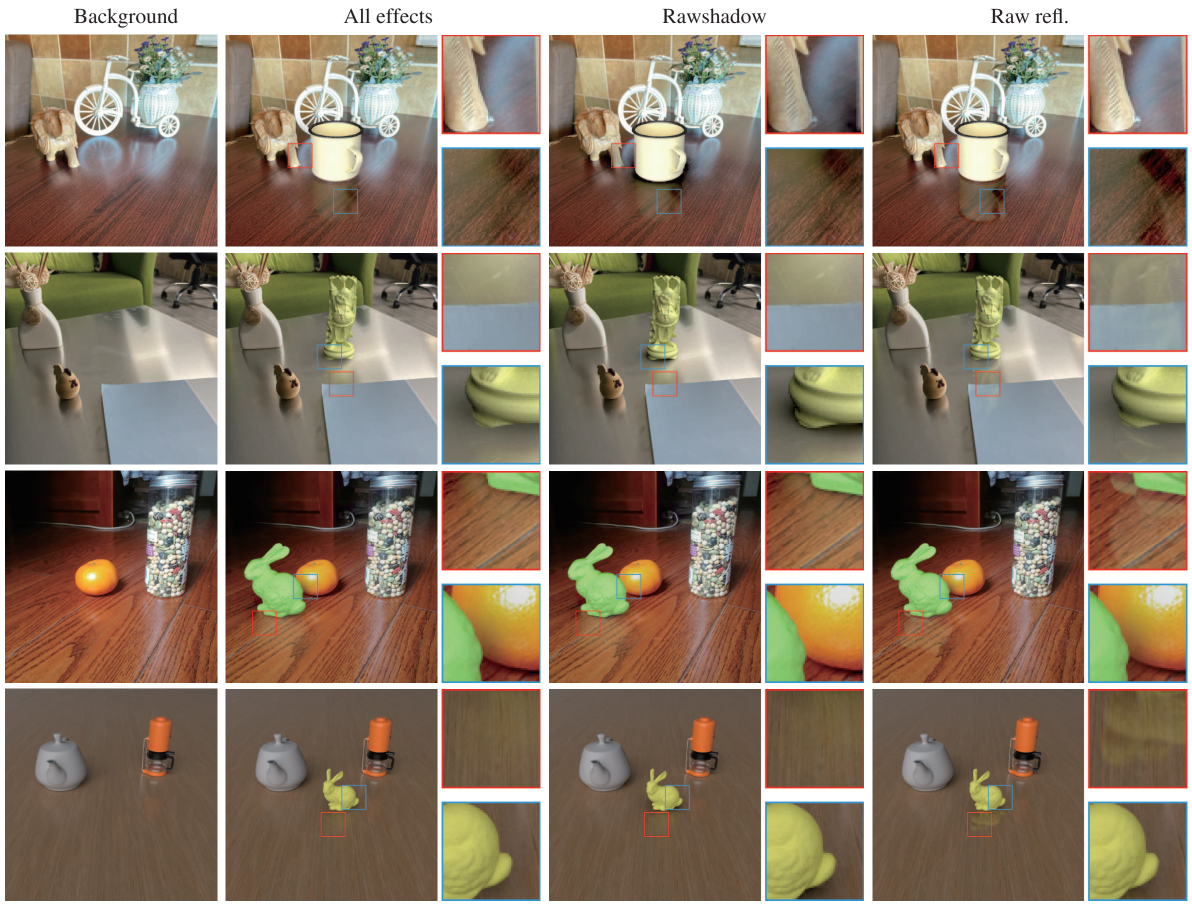

Figure 5 (Color online) Ablation study. The "All effects" column runs our full pipeline. The "Raw shadow" column disables the shadow network and blends with the raw shadow layer. The "Raw refl." column disables both the reflection network and the roughness network and adds the specular reflection directly. The background images in the first three rows are real-world photos while that in the fourth row is synthetically rendered.

4 Experimental results

We have implemented a real-time rendering system using OpenGL for rendering and GPU-based TensorFlow for neural compositing. Running on a NVIDIA GTX 1050 Ti notebook GPU, our system can render a dynamic AR scene on a fixed background image of 1280×720 at 25 fps. The light and roughness networks run only once for each image taking about 28 ms. Layer generation as well as shadow and reflection networks forward propagation is performed each frame taking about 40 ms in total, where OpenGL pipeline takes about 12 ms and running networks takes about 23 ms. The rest time is mainly for data transfer between CPU and GPU. Please see the live demo in the supplementary video.

Ablation study. We performed ablation studies both quantitatively on a synthetic dataset and qualitatively on real photos to demonstrate the effects of our shadows and reflection networks.

The synthetic dataset contains 3000 images generated with the same method for training dataset and is unseen to the neural networks. We use average per-pixel L1 loss and average per-pixel L2 loss in image space to measure differences between final compositing and ground truth image. Table 2 demonstrates the full pipeline with refined shadow and reflection gives the best compositing result.

The results on real photos are given in Figure 5. As illustrated in the top row and the last row,

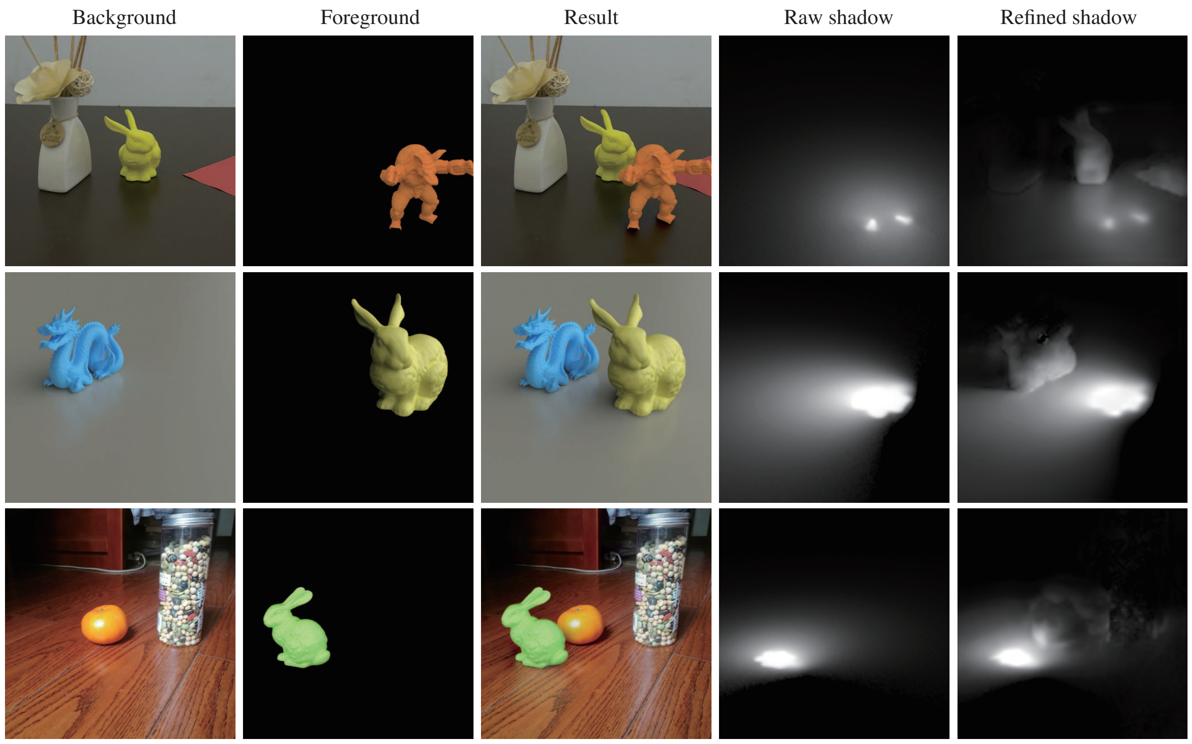

Figure 6 (Color online) Visualization of raw shadow layers and refined shadow layers. The "Foreground" column shows the rendered virtual object, the "Result" column shows the compositing result of our pipeline, the "Raw shadow" column shows the directly added shadow layers, and the "Refined shadow" shows the refined shadow layers by our network. A gamma correction ($\gamma = 2.2$) is applied to both layers to better visualize details. Shadow on glossy surface is slightly reduced by the shadow network. Note that the occlusion on non-plane background objects is also learnt automatically by our network.

the shadow network removes excessive diffuse PRT shadow on the reflective table and reduces excessive occlusion on the nearby non-planar elephant leg. On the other cases, as illustrated in the third row of Figure 5, the shadow network augments occlusion and learns to cast shadow to real objects. Figure 6 provides evidence that the shadow network learns a non-trivial representation of the scene geometry. In all examples, the roughness and reflection networks successfully imitate the blurring characteristics of real objects while the raw rendering looks excessively clear. In the second row, the networks also emulate the spatial variance of real reflectivity. More results can be found in the supplementary video.

We also prove the necessity of roughness network in Figure 7, where we compare final compositing with and without reflection mask. These results demonstrate that the reflection mask dims reflection on matte materials and contributes to the physical correctness of final compositing. More results of reflection mask on real photos are put in Figure S2 of Appendix A.

Comparison. We compare our rendering results with ground truth images in two cases, one real scene and one synthetic scene. In the real scene (Figure 8), two photos were captured with and without a 3D-printed bunny, and a virtual bunny was rendered into the background image. In the synthetic scene (Figure 9), a physically-based renderer, Mitsuba, was used to render the background and reference images. In both cases, our method generates realistic results with plausible shadow and reflection. On the other hand, our shadow and shading are less sharp owing to the low-frequency lighting assumption.

Figure 10 compares our method with the study of Karsch et al. [13], a state-of-the-art offline method for rendering synthetic objects into photographs. Both methods are automatic. Our real-time shading of the virtual Lucy is less sophisticated than their offline method. However, our method correctly detects the floor as reflective surface and produces plausible reflection without any user interaction.

Figures 11 and 12 compare our method with pure lighting reconstruction systems like DeepLight [18] and the work of Garon et al. [20] as well as harmonization method like Tsai et al. [9]. To make Tsai et al. [9] comparable with ours, we render the foreground objects with a constant uniform lighting and run their network to get the results. It is noteworthy that image harmonization merely adjusts the foreground and is not capable of adding felicitous shadows and reflections. Using a simplified network trained on less data and focused on low-frequency lighting, our lighting estimation is not as expressive as DeepLight.

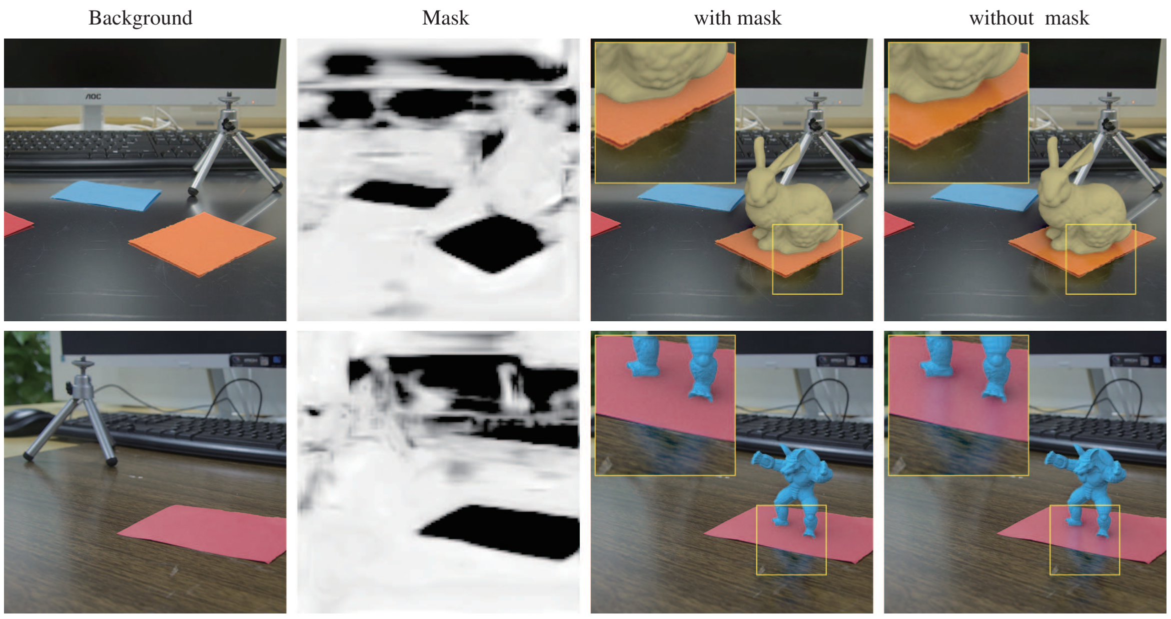

Figure 7 (Color online) Ablation study of reflection mask. Background images and reflection masks are shown in the first column. Compositing results with and without reflection masks are shown in the second and the third columns.

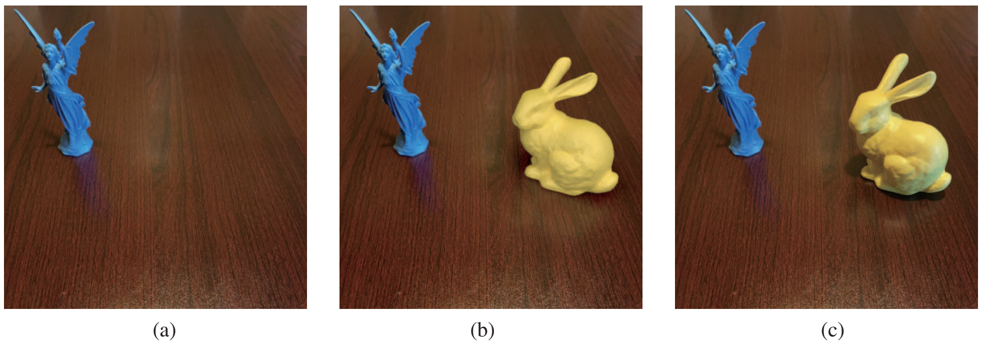

Figure 8 (Color online) Comparison with ground truth (a real photo). (a) Background; (b) our result; (c) real photo.

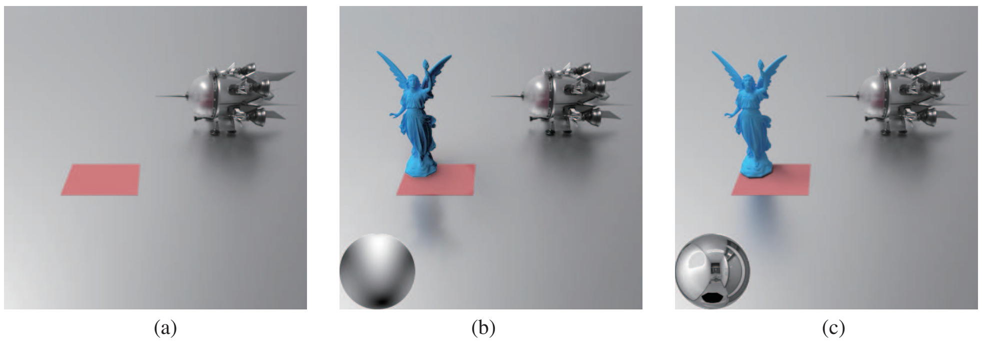

Figure 9 (Color online) Comparison with ground truth (a synthetic image). The estimated lighting and the ground truth lighting are also shown as insets. (a) Background; (b) our result; (c) reference.

However, we are able to produce reflections that resemble the ground truth photograph. More comparison results are given in Figure S3 of Appendix A.

We also try inserting multiple virtual objects into a scene. Though trained with only one foreground object, the networks can still generate reasonable results, as shown in the last row of Figure 12. We attribute it to the guidance of raw shadow and reflection generated by the renderer, which simplifies the task into a more localized and easier one for neural networks to handle. Occlusions and inter-reflections among virtual objects are not included in our implementation of real-time layer generation. The artifacts

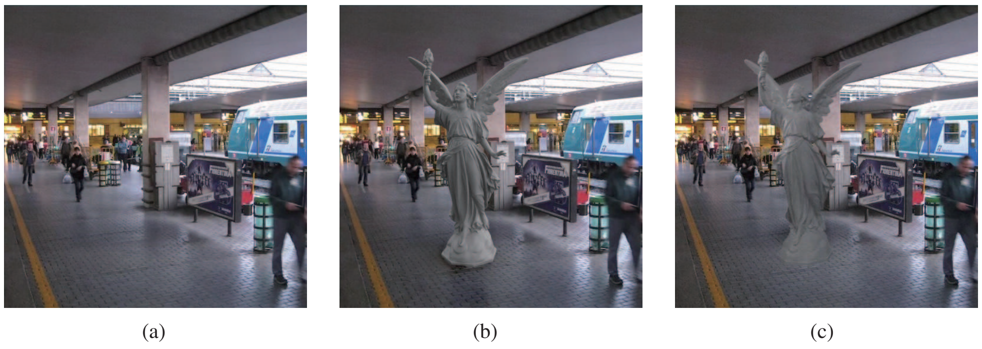

Figure 10 (Color online) Comparison with the work of Karsch et al. [13].

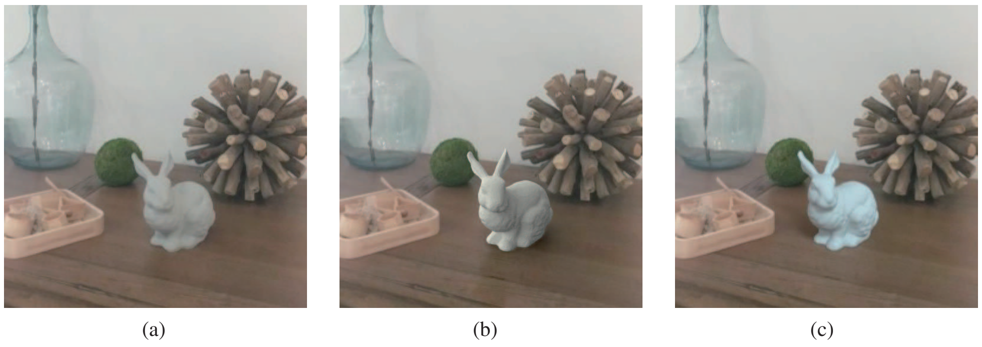

Figure 11 (Color online) Comparison with DeepLight [18]. The bunny is virtual in (b) and (c).

are often not noticeable in low-frequency lighting environments unless the virtual objects get close enough to each other. Handling these inter-reflections between virtual objects or between real and virtual objects is to be solved in future work.

User study. In order to evaluate the quality of composed images, we perform a user study in the form of online questionaires and received 50 valid questionaires (22 females and 27 were familiar with or had ever used an augmented reality application). The age range is 15–53 (average: 30.7, median: 26.5) in the valid questionaires. We create a test set of 15 real photographs (11 indoor scenes and 4 outdoor scenes) and show the subjects compositing results (see Figure S3 in Appendix A) of our method, Tsai et al. [9] and Garon et al. [20]. The compositing images of each scene were randomly shuffled when presented to subjects. The participants were allowed to watch images of each scene only once, to evaluate their first impression. The participants were asked to rate them based on two criteria, consistency between the virtual object and the background (e.g., position and color of highlight, shadow and reflection) and visual appeal. The score ranges from 1 to 5 corresponding poor to good.

The results are illustrated in Table 3. The results show that our scores are approximate 0.2 higher than Garon et al. [20] and are much higher than Tsai et al. [9] at both criteria. It indicates that with reflections and improved shadows, our method can bring more reality to the compositing results.

While trained only on synthesized tabletop scenes with wood-like materials, our method can plausibly generalize to scenes with a variety of scales, background objects and surface materials. As illustrated in Figure S1 of Appendix A, our method can infer reflectivity on marble, painted and water surfaces, and can handle room-scale to building-scale photographs.

5 Conclusion and future work

We have presented a deep-learning based method for AR rendering with shadow and reflection effects. Our key contribution is the neural compositing framework that uses CNNs to composite rendered layers of virtual objects with real images to emulate physically sophisticated effects. We believe that our work can serve as a useful alternative to the traditional reconstruct-then-render approaches, and expect it could inspire further research along this direction.

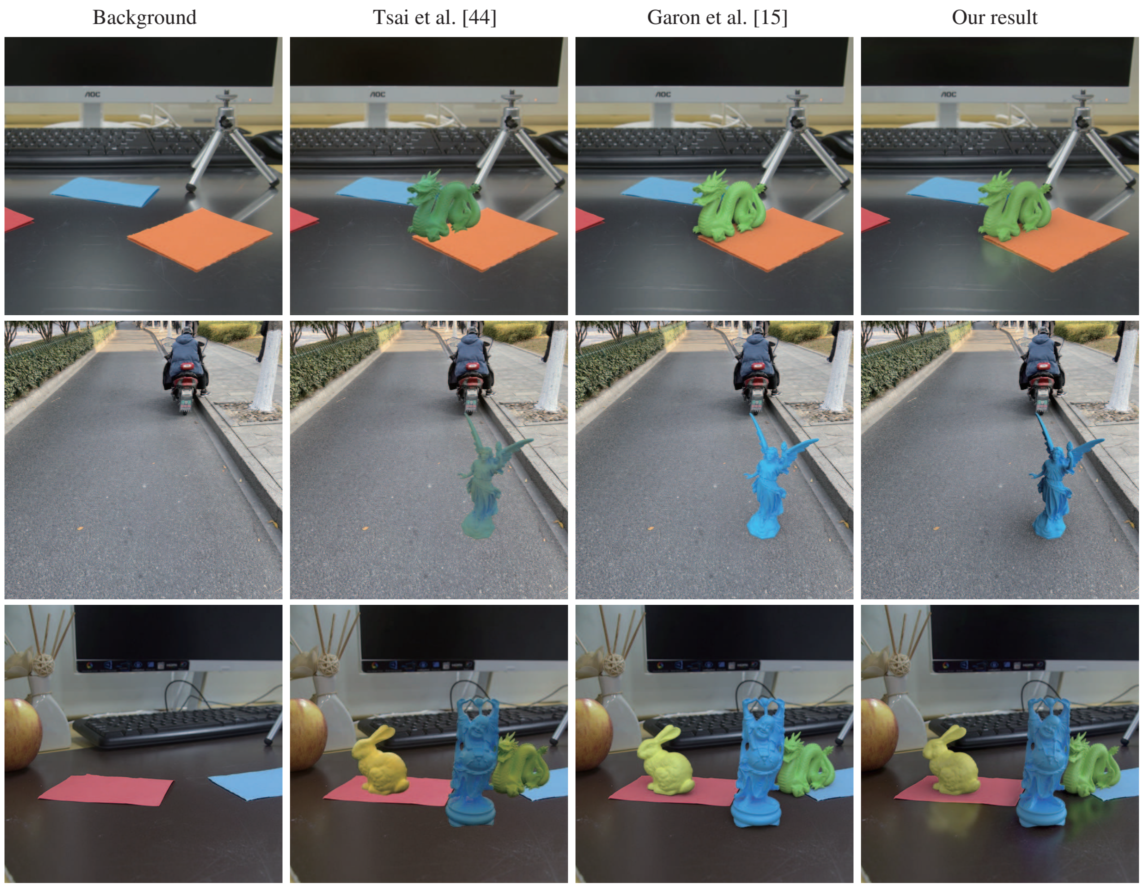

Figure 12 (Color online) Comparisons with Tsai et al. [9], Garon et al. [20]. The virtual objects of the first two rows are the dragon and lucy. Multiple virtual objects (bunny, dragon and buddha) are inserted in the last row to valify our method's capacity in a more general case.

Table 3 The user study results[a)]

	Tsai et al. [9]	Garon et al. [20]	Our method
Consistency	2.452	3.475	**3.662**
Visual appeal	2.641	3.522	**3.755**

a) The participants were asked to rate images according to consistency between the virtual object and the background and visual appeal. The average scores are shown in the table with scores (1–5) indicating poor to good. The best scores are highlighted.

Currently, our system only processes a single image and requires a desktop GPU to run the neural networks in real time. However, handling live video input on a mobile device should be within reach on next-generation mobile devices. Our system already restricts its input to what mobile AR SDKs can provide, while significant acceleration can be expected from mobile accelerator chips and network optimization techniques. In fact, DeepLight has already demonstrated real-time frame rates on mobile CPUs [18].

We currently focus on the more regular floor reflection instead of general inter-reflection light paths. It could be worthy to explore non-image layer representations where sophisticated paths could be more easily post-processed. Also, our method still has a considerable reliance on the light estimation quality. Problematic light network outputs can lead to shading inconsistency between virtual and real objects, as illustrated in Figure 13(a).

Another direction is to extend to high-frequency lighting environments. High frequency lighting estimation and PRT methods have already been established [18, 43]. However, rendering under such lighting conditions tends to generate sharp features, challenging our current networks. The networks also fail to generate compatible sharp reflections on some low roughness surface even in low-frequency lighting environments, shown in Figure 13(b).

Our method works well in most cases but some complicated interactions still remain unconsidered

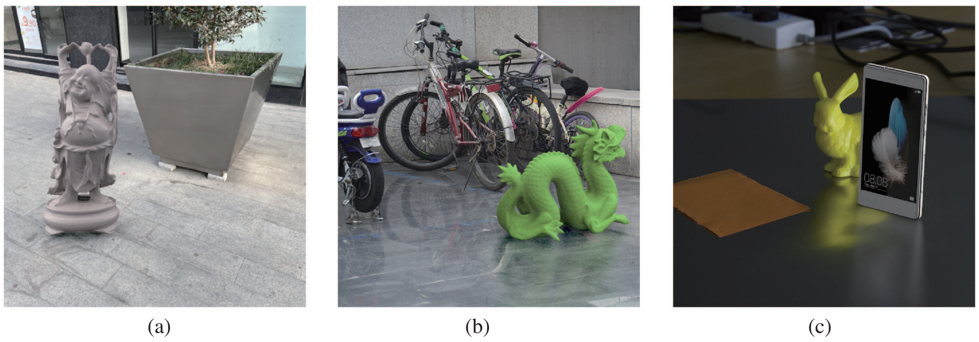

Figure 13 (Color online) From left to right: (a) problematic lighting estimation leads to inconsistent shading; (b) over blurring reflection caused by upsampling from the limited network resolution; (c) reflection overlapping artifact. The virtual objects are the buddha, dragon and mobile phone.

between a virtual object and a real scene. As shown in Figure 13(c), our method is unable to remove reflections on background. This is a common problem of rendering differential images and blending which is left for future work. As we assume the virtual objects are placed in the front of all real objects, occlusions are also not considered. This is also for future study.

Supporting information Appendix A. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Sloan P P, Kautz J, Snyder J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans Graph, 2002, 21: 527–536
2 Reinhard E, Adhikhmin M, Gooch B, et al. Color transfer between images. IEEE Comput Grap Appl, 2001, 21: 34–41
3 Pitie F, Kokaram A. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. In: Proceedings of the 4th European Conference on Visual Media Production, 2007. 1–9
4 Sunkavalli K, Johnson M K, Matusik W, et al. Multi-scale image harmonization. ACM Trans Graph, 2010, 29: 1–10
5 Pérez P, Gangnet M, Blake A. Poisson image editing. ACM Trans Graph, 2003, 22: 313
6 Tao M W, Johnson M K, Paris S. Error-tolerant image compositing. In: Proceedings of European Conference on Computer Vision. Berlin: Springer, 2010. 31–44
7 Johnson M K, Dale K, Avidan S, et al. CG2Real: improving the realism of computer generated images using a large collection of photographs. IEEE Trans Visual Comput Graph, 2011, 17: 1273–1285
8 Lalonde J F, Efros A A. Using color compatibility for assessing image realism. In: Proceedings of 2007 IEEE 11th International Conference on Computer Vision, 2007. 1–8
9 Tsai Y H, Shen X, Lin Z, et al. Deep image harmonization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 3789–3797
10 Cong W, Zhang J, Niu L, et al. Dovenet: deep image harmonization via domain verification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 8394–8403
11 Debevec P. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. New York: Association for Computing Machinery, 1998. 189–198
12 Agusanto K, Li L, Chuangui Z, et al. Photorealistic rendering for augmented reality using environment illumination. In: Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. 208–216
13 Karsch K, Sunkavalli K, Hadap S, et al. Automatic scene inference for 3D object compositing. ACM Trans Graph, 2014, 33: 1–15
14 Aittala M. Inverse lighting and photorealistic rendering for augmented reality. Vis Comput, 2010, 26: 669–678
15 Boivin S, Gagalowicz A. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 2001. 107–116
16 Hold-Geoffroy Y, Sunkavalli K, Hadap S, et al. Deep outdoor illumination estimation. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2017
17 Hold-Geoffroy Y, Athawale A, Lalonde J. Deep sky modeling for single image outdoor lighting estimation. 2019. ArXiv: 1905.03897
18 LeGendre C, Ma W C, Fyffe G, et al. Deeplight: learning illumination for unconstrained mobile mixed reality. In: Proceedings of ACM SIGGRAPH 2019 Talks. New York: Association for Computing Machinery, 2019
19 Song S, Funkhouser T. Neural illumination: lighting prediction for indoor environments. 2019. ArXiv: 1906.07370
20 Garon M, Sunkavalli K, Hadap S, et al. Fast spatially-varying indoor lighting estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 6908–6917
21 Sengupta S, Gu J, Kim K, et al. Neural inverse rendering of an indoor scene from a single image. In: Proceedings of International Conference on Computer Vision (ICCV), 2019
22 Li X, Dong Y, Peers P, et al. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. ACM Trans Graph, 2017, 36: 1–11
23 Li Z, Xu Z, Ramamoorthi R, et al. Learning to reconstruct shape and spatially-varying reflectance from a single image. In: Proceedings of SIGGRAPH Asia 2018 Technical Papers. New York: ACM, 2018. 269
24 Gao D, Li X, Dong Y, et al. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. ACM Trans Graph, 2019, 38: 1–15

25 Meka A, Maximov M, Zollhoefer M, et al. Lime: live intrinsic material estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 6315–6324

26 Karsch K, Hedau V, Forsyth D, et al. Rendering synthetic objects into legacy photographs. ACM Trans Graph, 2011, 30: 1–12

27 Kán P, Kaufmann H. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In: Proceedings of 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2012. 99–108

28 Kán P, Kaufmann H. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In: Proceedings of 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2013. 133–141

29 Meshry M, Goldman D B, Khamis S, et al. Neural rerendering in the wild. 2019. ArXiv: 1904.04290

30 Thies J, Zollhöfer M, Nieundefinedner M. Deferred neural rendering: image synthesis using neural textures. ACM Trans Graph, 2019, 38: 1–12

31 Li T M, Aittala M, Durand F, et al. Differentiable Monte Carlo ray tracing through edge sampling. ACM Trans Graph, 2019, 37: 1–11

32 Che C, Luan F, Zhao S, et al. Inverse transport networks. 2018. ArXiv: 1809.10820

33 Zhang C, Wu L, Zheng C, et al. A differential theory of radiative transfer. ACM Trans Graph, 2019, 38: 1–16

34 Loper M M, Black M J. Opendr: an approximate differentiable renderer. In: Computer Vision — ECCV 2014. Berlin: Springer, 2014. 154–169

35 Kato H, Ushiku Y, Harada T, et al. Neural 3D mesh renderer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018

36 Dobashi Y, Iwasaki W, Ono A, et al. An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. ACM Trans Graph, 2012, 31: 1–10

37 Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention — MICCAI 2015, Berlin: Springer, 2015. 234–241

38 Walter B, Marschner S, Li H, et al. Microfacet models for refraction through rough surfaces. In: Proceedings of Eurographics Symposium on Rendering, 2007. 195–206

39 Gardner M A, Sunkavalli K, Yumer E, et al. Learning to predict indoor illumination from a single image. ACM Trans Graph, 2017, 36: 1–14

40 Wald I, Woop S, Benthin C, et al. Embree-a ray tracing kernel framework for efficient CPU ray tracing. ACM Trans Graph, 2014, 33: 1–8

41 Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations, San Diego, 2015

42 Bojanowski P, Joulin A, Lopez-Paz D, et al. Optimizing the latent space of generative networks. 2017. ArXiv: 1707.05776

43 Ng R, Ramamoorthi R, Hanrahan P. All-frequency shadows using non-linear wavelet lighting approximation. In: Proceedings of ACM SIGGRAPH 2003 Papers. New York: Association for Computing Machinery, 2003. 376–381